APPENDIX A

```
/*
 * /-------------------------------------------------------------------------\
 * | The following programs are the sole property of Avid Technology, Inc., |
 * | and contain its proprietary and confidential information.              |
 * |                   Copyright © 1989-1992 Avid Technology, Inc.           |
 * \-------------------------------------------------------------------------/
 *
 * Module Name:         mfm_allocate.c
 *
 * Module Description:
 *
 */
#include    "mfm_allocate.h"

#include    "AvidGlobals.h"
#include    "expansionDefs.h"
#include    "LinkList.h"
#include    "mfm.h"
#include    "disk_mac.h"
#include    "memrtns.h"
#include    "Digitize.h"
#include    "LogicalToPhysical.h"
#include    "channel.h"
#include    "ResourceBible.h"
#include    "env.h"
#include    "uid.h"
#include    "MacUtils.h"
#include    "DebugUtils.h"
#include    "VolumeMenu.h"
#include    "JPEGUtils.h"
#include    "Exception.h"
#include    "dialogUtils.h"
#include    "FSutils.h"
#include    "BaseErrorDefs.h"
#include    "autorequest.h"
#include    "ResourceDefs.h"
#include    "videoDefs.h"

#define BREATHING_ROOM  200     /* KB to leave for directory expansion */

#define DIG_MODE    1
#define LOG_MODE    2

typedef struct
    {
    MFM_CRUX    crux;
    short       vRef;
    channel_t   channel;
    long        bytesPerSec;
    long        blocksToAlloc;
    long        blockSize;
    } mfm_alloc_t,
    * mfm_alloc_ptr,
    **mfm_alloc_hdl;

/***** Static Variables *****/
static  listID          alloc = NIL;
static  u_long          ApproxFrameSize = 1L;
static  char            theCapMode      = DIG_MODE;          // DIG_MODE, LOG_MODE
static  Ftype_t         theFtype        = 0;
static  float           theCapRate      = 0;
static  MFM_CRUX        theVcrux        = 0;                 // When these are zero the cruxes are clo
static  MFM_CRUX        theA1crux       = 0;
static  MFM_CRUX        theA2crux       = 0;
static  short           theVvref        = BAD_VREFNUM,
                        theAvref        = BAD_VREFNUM;
static  long            theSampsPerSec  = 0,
                        theBytesPerSamp = 0;
static  long            theTimeAvail    = 0;                 // the minimum of the times available in
static  videoFormat_t   theVideoFormat  = PAL_f;                    // more likely to catch bugs by initing t
static  sourceFormat_t  theSourceFormat = VIDEO_f;
static  videoType_t     theVideoType    = {8, VMHiResHiColor};      // HACK for now
static  capture_mask_t  theCaptureMask  = 0L;
static  capture_mask_t  theResultMask   = 0L;
static  u_char          theCapShift     = 0;
static  u_char          theResultShift  = 0;

static  channel_t       theChannels     = 0;
static  audioClock_t    theAudioClock   = Clock44100;
static  audioRate_t     theAudioRate    = halfRate;
static  Boolean         theAudioMixed   = FALSE;
static  Boolean         useEmptiestVideo = TRUE,
                        useEmptiestAudio = TRUE;

/***** Defined Below *****/
static  void            setVinfo( Ftype_t Ftype, capture_mask_t captureMask, u_char captureShift, float captu
static  void            amMItem2Val( short mitem, audioClock_t *audioClock, audioRate_t *audioRate, long *aud
static  MFM_CRUX        mfaAllocCreate(long bytesPerSec, short vref, channel_t channel, Boolean preflight);
static  void            mfaAllocCalc(Boolean preflight);
static  void            mfaAllocEnd(void);
static  void            mfaAllocPunt(void);
static  long            TotalBytes(short theVref);
static  void            checkVolumeSettings(void);
static videoModifier_t  getVideoModifier(short lQuality, short cQuality);

/*****************************************************/
/****      Public Code                           ****/
/*****************************************************/


/*************************************************************************************
 *
 *  mfaSetSettings
 */
Boolean mfaSetSettings( channel_t chans, float capRate, u_char phase,
                        audioClock_t audioClock, audioRate_t audioRate, Boolean audioMixed,
                        short Vvref, short Avref, videoType_t video_type)
    {
    Boolean     needsReinit;

    mfaForgetFiles (OUT_ALL);
    needsReinit = FALSE;

    if(!CksumValid[ck_44khz] && audioRate == fullRate)
        audioRate = halfRate;
    if(!CksumValid[ck_48khz])
        audioClock = Clock44100;

    if (theCapMode == DIG_MODE && (theChannels != chans ||
                                   theCapRate != capRate ||
                                   theAudioClock != audioClock ||
                                   theAudioRate != audioRate ||
                                   theAudioMixed != audioMixed))
            needsReinit = TRUE;

    /*
     *  Set the mfm_allocate statics
     */

    theChannels     = chans;
    theCapRate      = capRate;
    theAudioClock   = audioClock;
    theAudioRate    = audioRate;
    theAudioMixed   = audioMixed;
    theVvref        = Vvref;
    theAvref        = Avref;

    useEmptiestVideo = (theVvref == BAD_VREFNUM);
```

```
useEmptiestAudio = (theAvref == BAD_VREFNUM);

xprotect
    {
    checkVolumeSettings ();
    }
xexception
    {
    if (!xcodeEquals (MFA_NO_MEDIA_DRIVES))
        xpropagate();

    auto_request("You will not be able digitize until a valid\nmedia volume is placed online.", "OK", 1);
    }
xend;

/*
*   Setup video capture mode info
*/
switch( (int)(theCapRate*10))
    {
    case 240:
        if( phase == 0)
                setVinfo (FULL, 0xD8000000L, 0, 24.0, 0x80000000L, 0);          // 1101 1xxx        ,4 ou
        else if( phase == 1)
                setVinfo (FULL, 0xB8000000L, 0, 24.0, 0x80000000L, 0);          // 1011 1xxx        ,4 ou
        else if( phase == 3)
                setVinfo (FULL, 0x78000000L, 1, 24.0, 0x80000000L, 0);          // 0111 1xxx        ,4 ou
        else
                setVinfo (FULL, 0xE8000000L, 0, 24.0, 0x80000000L, 0);          // 1110 1xxx        ,4 ou
        break;
    case 120:
        if( phase == 0 || phase == 1)
                setVinfo (FULL, 0x48000000L, 2, 12.0, 0x40000000L, 1);          // 0100 1xxx        ,2 ou
        else
                setVinfo (FULL, 0x28000000L, 2, 12.0, 0x40000000L, 1);          // 0010 1xxx        ,2 ou
        break;
    case  60:   setVinfo (FULL, 0x08000000L, 4,  6.0, 0x10000000L, 3);  break;  // 0000 1xxx        ,1 ou
    case 300:   setVinfo (FULL, 0x80000000L, 0, 30.0, 0x80000000L, 0);  break;  // 1xxx xxxx        ,1 ou
    case 150:   setVinfo (FULL, 0x40000000L, 1, 15.0, 0x40000000L, 1);  break;  // 01xx xxxx        ,1 ou
    case 100:   setVinfo (FULL, 0x20000000L, 2, 10.0, 0x20000000L, 2);  break;  // 001x xxxx        ,1 ou
    case 250:   setVinfo (FULL, 0x80000000L, 0, 25.0, 0x80000000L, 0);  break;  // 1xxx xxxx        ,1 ou
    case 125:   setVinfo (FULL, 0x40000000L, 1, 12.5, 0x40000000L, 1);  break;  // 01xx xxxx        ,1 ou
    case  50:   setVinfo (FULL, 0x08000000L, 4,  5.0, 0x08000000L, 4);  break;  // 0000 1xxx        ,1 ou
    }

theSourceFormat = sourceFormat;                                                 // Get it from global no :
theVideoFormat = videoFormat;                                                   // Get it from global no :
theVideoType.vcID = gVideoType.vcID;                                            // Get it from global no :
theVideoType.videoModifier = video_type.videoModifier;

SetDigitizeCaptureMask (theCaptureMask, theCapShift);

/*
*   Setup audio capture mode info
*/
theSampsPerSec      = ((audioRate == fullRate) ? (audioClockToClockRate(audioClock)) : (audioClockToClock
theBytesPerSamp     = (audioRate == fullRate ? 2 : 1);

return needsReinit;
}
```

```
* Addresses of hardware registers:
divect       .set    0fffffea0h  ;the Display-Interrupt vector location
dpytrap      .set    0fffffea0h  ;address of DPYINT trap vector
mode         .set    0f8600000h  ;video mode register
status       .set    0f8290000h  ;video status register

vsblnk       .set    0c0000060h  ;gsp control registers:
vtotal       .set    0c0000070h  ;total vertical lines
dpyctl       .set    0c0000080h  ;
dpystrt      .set    0c0000090h  ;
dpyint       .set    0c00000a0h  ;
control      .set    0c00000b0h  ;
hstctl1      .set    0c00000f0h  ;
intenb       .set    0c0000110h  ;
intpend      .set    0c0000120h  ;
convsp       .set    0c0000130h  ;
convdp       .set    0c0000140h  ;
psize        .set    0c0000150h  ;
pmask        .set    0c0000160h  ;
pmaskext     .set    0c0000170h  ;

* Constants and masks:
msginmsk     .set    0007h       ;Fields in hstctl1 register
msgoutmsk    .set    0070h
msginisave   .set    0002h
msgindbg     .set    0007h
msgintmsk    .set    0008h       ;intin field in hstctl1
msginf2      .set    0003h
msgoutisave  .set    0020h
msgoutdbg    .set    0070h
msgoutf2     .set    0030h
msgoutinc    .set    0010h
intin        .set    0008h
intout       .set    0080h
ctlmsk       .set    801fh       ;Mask for the CONTROL register.
di           .set    10          ;Bit number of Display Interrupt bit
dispint      .set    1<<di       ;"Display Interrupt" bit of intenb and intpend
ni           .set    14          ;Bit number of Non Interlaced bit
notinterl    .set    1<<ni       ;The non-interlaced bit
ce_bit       .set    8000h       ;"Capture Enable" bit of video mode register
di           .set    10          ;Bit number of Display Interrupt bit
special      .set    2000000h    ;Offset for special JPEG hardware fifo "memory space"
palLines     .set    576         ;Number of lines in a frame
ntscLines    .set    480         ;Number of lines in a frame
rowbase      .set    0f8000000h  ;row table main picture starting address
traps        .set    0fffffc00h  ;address of trap page
macrows      .set    480         ;mac row table entries
vrows        .set    palLines+8  ;video rows in row table (incl color table & PAL)
frows        .set    palLines/2  ; Maximum # lines in a field (pal is larger)
maxField     .set    palLines/2  ; maximum # of lines in a field
pmemrow      .set    8000h       ;length in bits of physical memory rows
NVBLBIT      .set    4           ;Not Vertical Blanked -- bit position in video status reg
dpitch       .set    4000h       ;pitch of MAC (16-bit pixel) lines (2 kB)
dpShift      .set    14          ;Shifting a number by this multiplies by dpitch
pixsize      .set    16          ;Pixel size constant for "psize" register
pstride      .set    64          ;Number of bits between pixel "hits" in output image

cmdNone          .set    0       ;undefined command code
cmdPlay          .set    1       ;normal multi-frame playback to alternate screen buffer
cmdPack          .set    2       ;(UNUSED in FullRes)Pack 256*192 image
cmdUnpack        .set    3       ;unpack still frame to vcopy double buffer area (decompress)
cmdShow          .set    4       ;unpack and show a still frame in main screen buffer
cmdFull          .set    5       ;full-screen playback on an NTSC monitor
cmdBigPack       .set    6       ; pack a 640x480 image
cmdBigUnpack     .set    7       ; unpack an image to 640*480
cmdUnpackAdd     .set    8       ; unpack and combine an image
cmdUnpack16      .set    9       ; unpack a 16 bit frame in 32 bit mode
cmdPack16        .set    10      ; pack a 16 bit frame in 32 bit mode
```

```
vramBase     .usect   "vector
frameBuf     .usect   "vectors",32
bigBuf       .usect   "vectors",32




* Routine to sync to an odd field:
syncodd:
s1   move     *Rstatp,Rtemp
     btst     0,Rtemp
     jrnz     s1
     move     *Rstatp,Rtemp
     btst     0,Rtemp
     jrnz     s1
s2   move     *Rstatp,Rtemp
     btst     0,Rtemp
     jrz      s2
     move     *Rstatp,Rtemp
     btst     0,Rtemp
     jrz      s2
     rets

* Routine to sync to an even field:
synceven:
s3   move     *Rstatp,Rtemp
     btst     0,Rtemp
     jrz      s3
     move     *Rstatp,Rtemp
     btst     0,Rtemp
     jrz      s3
s4   move     *Rstatp,Rtemp
     btst     0,Rtemp
     jrnz     s4
     move     *Rstatp,Rtemp
     btst     0,Rtemp
     jrnz     s4
     rets
```

```
MEMORY
{
    MAPPE2: origin = 0ffd00000h,    length = 200000h
    NOMAP:  origin = 0fff00000h,    length = 0c8000h
    JSTAT:  origin = 0f8e00000h,    length = 16
    VEC:    origin = 0ffffd000h,    length = 000100h
}

SECTIONS
{
    vectors:      {} > VEC
    args:         {} > NOMAP
    .data:        {} > NOMAP
    .text:        {} > MAPPE2
    jstatus:      {} > JSTAT
}
```

```
    .title  "VISTA image capture and compress"

 * /----------------------------------------------------------------------\
 * | The following programs are the sole property of Avid Technology, Inc., |
 * | and contain its proprietary and confidential information.              |
 * |                     Copyright © 1989-1991 Avid Technology, Inc.        |
 * \----------------------------------------------------------------------/

* General register names:
Rtemp    .set    A0  ;Temp register
Rpixcnt .set    A1  ;Constant Pixels per line
Rpixel   .set    A2  ;Pointer to current input pixel
Rpxincl .set    A3  ;Constant # of bits to increment Rpixel to next input pixel
Rpxinc2 .set    A4  ;Alternate Constant to increment Rpixel to next input pixel
Rline    .set    A5  ;Constant Pitch of an input line in bits (same value as Sptch)
Rpixtmp .set    A6  ;Temp register for writing to pixel locations
Rjstatp .set    A7  ;Constant pointer to JPEG fifo status
Rx       .set    A8  ;Counter of pixels in a line
Rnext    .set    A9  ;Pointer to next input line
Rstatp   .set    A10 ;Constant pointer to video status
Rblack   .set    A11 ;Pointer to a black pixel
Rtemp2   .set    A12
R13      .set    A13
R14      .set    A14

Saddr    .set    B0  ;Source pixel array starting address
Sptch    .set    B1  ;Source pitch (# of bits from one line to next)
Offset   .set    B4  ;Base address of source pixel array
Bxy      .set    B7  ;Pixel array dimensions(rows:columns)
Rlincnt .set    B9  ;Constant: lines per frame
Ry       .set    B10 ;Counter: lines per frame
Rcapture .set   B11 ;Bit mask: frame skipper
Rloadcap .set   B12 ;Bit mask: used to reinit Rcapture
RB13     .set    B13
RB14     .set    B14

pixmsk   .set    8000h   ;Constant for "pmask" register (kill alpha chan)
spitch   .set    8000h   ;Constant for "Sptch" register (4 kBytes in bits)

         .copy   "equates.i"

jstatus .usect  "jstatus",16            ;JPEG fifo status

* Args TO <- and FROM -> the NuVista processor:
initcm   .usect  "args",32   ;<-initial capture mask
captmsk .usect  "args",32   ;<-reload capture mask
overrun .usect  "args",32   ;->number of overruns detected (inited by Mac)
frames   .usect  "args",32   ;->number of frames seen (inited by Mac)
tdummy1 .usect  "args",32   ;"fence" arg in other µcode <-
tdummy2 .usect  "args",32   ;"fencerr" arg in other µcode ->
tx       .usect  "args",32   ;<-number of x locs to hit
ty       .usect  "args",32   ;<-number of y locs (lines) to hit
tstride1 .usect "args",32   ;<-stride in bits between input pixel locs
tstride2 .usect "args",32   ;<-alt stride in bits between input pixel locs
tdelay   .usect  "args",32   ;<-amount of delay before capturing each line (default = 1)

         .copy   "captureMacros.i"

    .data
stack:   .bes    4000h   ;Stack space (2kB) for calls and interrupts

    .page
    .text
    .align

Flag:
    .long   0                ; Debug: Current value of pixel fifo status
Dat:
    .long   0,0,0,0,0,0,0,0 ; Reserved for debug info
```

```
* Start of main program

    .def    _main
_main
    setf    16,0,0          ; Field zero is 16-bit unsigned
    setf    32,0,1          ; Field one is 32-bit unsigned

    movi    stack,sp        ; Load stack pointer

    movi    spitch,Sptch    ; Load contstant number of bits per line
    move    Sptch,Rline
    movi    pixmsk,Rtemp    ; Init pixel mask
    move    Rtemp,@pmask
    move    Rtemp,@pmaskext
    movi    jstatus+8,Rjstatp ; Load pointer to JPEG status register
    clr     Rpixtmp         ; Clear pixel temp
    movi    status,Rstatp   ; Load pointer to video status register

*   Clear DONE and wait for GO:
    clr     Rtemp
    movb    Rtemp,@hstctll  ;clear msgout (the DONE bit and interrupt bits) to host
hosths:
    movb    @hstctll,Rtemp  ;read host control register
    andi    msginmsk,Rtemp  ;mask message
    jrz     hosths          ;wait for GO signal (any non-zero value)
    movi    msgoutinc,Rtemp
    move    Rtemp,@hstctll  ;set indicator to let host know we have started

*   Get some host args into registers:
    move    @tx,Rpixcnt,1       ;number of stores in x
    move    @ty,Rlincnt,1       ;number of lines in frame
    move    @tstride1,Rpxinc1,1 ;number of bits between pixels
    move    @tstride2,Rpxinc2,1 ;alt number of bits between pixels

*   For debug, write parameters back to memory:
    movi    Dat,Rtemp           ;get addr of debug dump area
    move    Rpixcnt,*Rtemp+,1   ;x
    move    @ty,*Rtemp+,1       ;y
    move    Rpxinc1,*Rtemp+,1   ;stride 1
    move    Rpxinc2,*Rtemp+,1   ;stride 2
    move    Rline,*Rtemp+,1     ;source pitch in bits (number of bits from one line to the next)

* N.B. The x argument (Rpixcnt) MUST be a multiple of 32!
    srl     5,Rpixcnt           ;divide line length (x) by 32 for unrolled loop

    callr   syncodd             ;FIRST TIME:  Wait for start of odd field

    move    @mode,Rtemp
    ori     ce_bit,Rtemp        ;set the global capture enable bit (begins digitizing)
    move    Rtemp,@mode

    move    @initcm,Rcapture,1  ;load initial capture mask
    move    @captmsk,Rloadcap,1 ;load value to reinitialize capture mask

    movi    black-special,Rblack    ;address of black ("0")

    jruc    frame

black:
    .long   0,0

    .align                      ;align the following code with the I-cache
skpfram:                        ;come here to skip capturing a frame
    callr   synceven
    callr   syncodd

* Attempt capturing a frame:
```

```
frame:
* Count the frame (N.B. We must count every frame seen, whether captured or skipped):
    move    @frames,Rtemp,1
    addk    1,Rtemp             ;count
    move    Rtemp,@frames,1

* Decide whether this is a frame we want, based on capture mask:
    sll     1,Rcapture          ;check next mask bit (it goes to C-bit)
    jrnc    skpfram             ;skip this frame if C-bit is zero (last active bit guaranteed to be 1)
    jrnz    mskok               ;check if need to reload mask bits: yes->fall thru
    move    Rloadcap,Rcapture   ;reload the capture mask (32 bits) for next time
mskok:
* Prepare for "lines" loop:
    move    @vramBase,Rpixel,1
    subi    special,Rpixel,1
*   movi    capture-special,Rpixel ;starting address of video frame bufr (Special space)
    move    Rpixel,Rnext        ;remember address of first line
    move    Rlincnt,Ry          ;get number of lines in frame

* Check video field (s/b ODD from compressing prev frame or from syncodd after hosths or skpfram).
* (N.B. Assumes compression takes more than one field time (~1/60th second), but less than a frame time.)
    callr   syncevck            ;wait for start of even field (i.e. digitizing complete)

* Add 8 lines of black to the top of the picture:
    movk    8,Rtemp2            ;eight groups of one line
blk
    move    Rpixcnt,Rx          ;pixels-per-line / 32
    sll     5-2,Rx              ;calc the loop count ( *32 ^ /4hits-per-loop)
loop2b:
    movb    *Rjstatp,Rtemp      ;read JPEG pixel fifo status
*   move    Rtemp,@Flag,0       ;***debug***
    jrnn    loop2b              ;wait until fifo ready (bit7 == 1)
blkloop
    move    Rpixtmp,*Rblack,0   ;each write causes auto xfer(s) to JPEG pixel fifo.
    move    Rpixtmp,*Rblack,0
    move    Rpixtmp,*Rblack,0
    move    Rpixtmp,*Rblack,0
    dsjs    Rx,blkloop          ;1 line of pixels
    dsjs    Rtemp2,blk

* Send frame interrupt to the Mac:
    move    @hstctl1,Rtemp      ;get hstctl1 value
    ori     intout,Rtemp        ;set interrupt bit
    move    Rtemp,@hstctl1      ;send to host to indicate frame start

* Start of loop to process all lines of a frame:
lines:
    add     Rline,Rnext         ;calc addr of next line
    move    Rpixcnt,Rx          ;(re)load x count (pixels-per-line / 32)

*   move    @tdelay,Rtemp2,1    ;DEBUG
*loop2d:
*   dsjs    Rtemp2,loop2d       ;DEBUG

loop2j:
    movb    *Rjstatp,Rtemp      ;read JPEG pixel fifo status
*   move    Rtemp,@Flag,0       ;***debug***
    jrnn    loop2j              ;wait until fifo ready (bit7 == 1)

loop2:
    move    Rpixtmp,*Rpixel,0   ;this write causes auto xfer(s) to JPEG pixel fifo.
    add     Rpxinc1,Rpixel      ;now advance to next pixel
    move    Rpixtmp,*Rpixel,0   ;2
    add     Rpxinc2,Rpixel
    move    Rpixtmp,*Rpixel,0   ;3
    add     Rpxinc1,Rpixel
    move    Rpixtmp,*Rpixel,0   ;4
    add     Rpxinc2,Rpixel
    move    Rpixtmp,*Rpixel,0   ;5
```

```
        add     Rpxincl,Rpixel
        move    Rpixtmp,*Rpixel,0    ;6
        add     Rpxinc2,Rpixel
        move    Rpixtmp,*Rpixel,0    ;7
        add     Rpxincl,Rpixel
        move    Rpixtmp,*Rpixel,0    ;8
        add     Rpxinc2,Rpixel
        move    Rpixtmp,*Rpixel,0    ;9
        add     Rpxincl,Rpixel
        move    Rpixtmp,*Rpixel,0    ;10
        add     Rpxinc2,Rpixel
        move    Rpixtmp,*Rpixel,0    ;11
        add     Rpxincl,Rpixel
        move    Rpixtmp,*Rpixel,0    ;12
        add     Rpxinc2,Rpixel
        move    Rpixtmp,*Rpixel,0    ;13
        add     Rpxincl,Rpixel
        move    Rpixtmp,*Rpixel,0    ;14
        add     Rpxinc2,Rpixel
        move    Rpixtmp,*Rpixel,0    ;15
        add     Rpxincl,Rpixel
        move    Rpixtmp,*Rpixel,0    ;16
        add     Rpxinc2,Rpixel
        move    Rpixtmp,*Rpixel,0    ;17
        add     Rpxincl,Rpixel
        move    Rpixtmp,*Rpixel,0    ;18
        add     Rpxinc2,Rpixel
        move    Rpixtmp,*Rpixel,0    ;19
        add     Rpxincl,Rpixel
        move    Rpixtmp,*Rpixel,0    ;20
        add     Rpxinc2,Rpixel
        move    Rpixtmp,*Rpixel,0    ;21
        add     Rpxincl,Rpixel
        move    Rpixtmp,*Rpixel,0    ;22
        add     Rpxinc2,Rpixel
        move    Rpixtmp,*Rpixel,0    ;23
        add     Rpxincl,Rpixel
        move    Rpixtmp,*Rpixel,0    ;24
        add     Rpxinc2,Rpixel
        move    Rpixtmp,*Rpixel,0    ;25
        add     Rpxincl,Rpixel
        move    Rpixtmp,*Rpixel,0    ;26
        add     Rpxinc2,Rpixel
        move    Rpixtmp,*Rpixel,0    ;27
        add     Rpxincl,Rpixel
        move    Rpixtmp,*Rpixel,0    ;28
        add     Rpxinc2,Rpixel
        move    Rpixtmp,*Rpixel,0    ;29
        add     Rpxincl,Rpixel
        move    Rpixtmp,*Rpixel,0    ;30
        add     Rpxinc2,Rpixel
        move    Rpixtmp,*Rpixel,0    ;31
        add     Rpxincl,Rpixel
        move    Rpixtmp,*Rpixel,0    ;32
        add     Rpxinc2,Rpixel
        dsj     Rx,loop2        ;loop thru the line

        move    Rnext,Rpixel    ;load addr of next line to process
        dsj     Ry,lines        ;loop for next line

*       callr   syncodd             ; If we're in odd field, it took too long. .
        jruc    frame

*
* The following routines sync the code to the incoming video fields.
* Note:  Since the status register is not synchronized with the 34010 instruction
*        clock, we must always check that we get the same reading twice in a row.
*
```

```
* Wait for start of next even field; check to make sure field is already ODD at entry.
* (If we enter here in an even field, it means an OVERRUN has occurred.)
syncevck:
s5       move     *Rstatp,Rtemp
         btst     0,Rtemp
         jrz      s5el              ; if even, go check a second time; fall thru if odd
s5ol     move     *Rstatp,Rtemp
         btst     0,Rtemp
         jrz      s5el              ; if even, go check a second time; fall thru if odd
s6       move     *Rstatp,Rtemp
         btst     0,Rtemp
         jrnz     s6                ; loop as long as it remains odd
         move     *Rstatp,Rtemp
         btst     0,Rtemp
         jrnz     s6                ; make sure we see it the same twice in a row
         rets                       ; normal successful return at start of an even field

* come here if we found an even value one time:
s5el     move     *Rstatp,Rtemp     ; perform second test for even
         btst     0,Rtemp
         jrnz     s5ol              ; jump back if second check is okay (odd)
* else, fall thru
* At this point we have an overrun (two evens in a row), so count it
         move     @overrun,Rtemp,1
         addk     1,Rtemp           ; In the even field already... increase overrun count
         move     Rtemp,@overrun,1
s7       move     *Rstatp,Rtemp     ; We know it is even, so now we need to wait for odd
         btst     0,Rtemp
         jrz      s7
         move     *Rstatp,Rtemp
         btst     0,Rtemp
         jrz      s7
         jruc     s6

     .end
```

APPENDIX 15

EDL

C

```
TITLE: UNTITLED01
FCM: NON-DROP FRAME
001 050     V     C          04:11:23:21 04:11:37:19  01:00:00:00 01:00:13:28
M2  050           030.0                  04:11:23:21
002 050     V     C          04:03:14:26 04:03:20:01  01:00:13:28 01:00:19:03
M2  050           030.0                  04:03:14:26
003 050     V     C          04:11:37:19 04:11:55:29  01:00:19:03 01:00:37:13
M2  050           030.0                  04:11:37:19
004 050     V     C          04:04:51:01 04:04:56:13  01:00:37:13 01:00:42:24
M2  050           030.0                  04:04:51:01
```

```
TITLE: UNTITLED01
FCM: NON-DROP FRAME
001  THEY_C   V     C          04:11:23:21 04:11:37:19 01:00:00:00 01:00:13:28
M2   THEY_C         030.0                  04:11:23:21
002  THEY_C   V     C          04:03:14:26 04:03:20:01 01:00:13:28 01:00:19:03
M2   THEY_C         030.0                  04:03:14:26
003  THEY_C   V     C          04:11:37:19 04:11:55:29 01:00:19:03 01:00:37:13
M2   THEY_C         030.0                  04:11:37:19
004  THEY_C   V     C          04:04:51:01 04:04:56:13 01:00:37:13 01:00:42:24
M2   THEY_C         030.0                  04:04:51:01
```

Avid Technology, Inc.

Assemble list for edl file picture:

```
  Seq               first edge                 last edge length       cum      Camera Roll
  ---              -----------                 --------- ------       ---     -------------
/-001          OPTICAL Number 1                  FADE IN   1+08      1+08        EFFECT
|
\-002      end of optical 1 to                 scene end   4+02      5+10        Flat #1

  003       KJ789876 -1370 +05                 -1372 +05   2+01      7+11        Flat #1

/-004            Scene start to     start of optical 2   1+04      8+15        Flat #1
|
| 005          OPTICAL Number 2                 DISSOLVE   3+00     11+15        EFFECT
|
\-006      end of optical 2 to                 scene end   7+05     19+04        Flat #1

  007       KJ789876 -1236 +02                 -1243 +09   7+08     26+12        Flat #1

/-008            Scene start to     start of optical 3   2+04     29+00        Flat #1
|
\-009          OPTICAL Number 3                 FADE OUT   1+08     30+08        EFFECT

  010         LEADER -0000 +00                 -0089 +15  90+00    120+08        LEADER

/-011          OPTICAL Number 4                  FADE IN   1+08    122+00        EFFECT
|
\-012      end of optical 4 to                 scene end   1+08    123+08        Flat #1

  013       KH123456 -5085 +05                 -5091 +10   6+06    129+14        Flat #1

  014       KJ789876 -1399 +05                 -1409 +08  10+04    140+02        Flat #1

  015         LEADER -0000 +00                 -0003 +14   3+15    144+01        LEADER

  016       KH123456 -5132 +02                 -5142 +04  10+03    154+04        Flat #1

  017       KH123456 -5053 +15                 -5057 +11   3+13    158+01        Flat #1

  018       KH123456 -5083 +00                 -5083 +13   0+14    158+15        Flat #1

  019       KJ789876 -1244 +09                 -1248 +09   4+01    163+00        Flat #1

  020       KJ789876 -1453 +07                 -1464 +11  11+05    174+05        Flat #1

/-021            Scene start to     start of optical 5   6+02    180+07        Flat #1
|
\-022          OPTICAL Number 5                 FADE OUT   1+08    181+15        EFFECT
```

Avid Technology, Inc.

Assemble Pull List (scene pull in assemble order) for edl file picture:

Tapename	Segment Name	first edge	last edge	length	scene
NAB91COMPILATIONTAPE	Flat #1	KJ789876 -1441 +15	-1575 +03	133+05	
NAB91COMPILATIONTAPE	Flat #1	KJ789876 -1368 +13	-1393 +07	24+11	
NAB91COMPILATIONTAPE	Flat #1	KH123456 -5019 +11	-5050 +04	30+10	
NAB91COMPILATIONTAPE	Flat #1	KJ789876 -1327 +03	-1368 +12	41+10	
NAB91COMPILATIONTAPE	Flat #1	KJ789876 -1234 +00	-1300 +00	66+01	
NAB91COMPILATIONTAPE	Flat #1	KH123456 -5050 +05	-5082 +15	32+11	
NAB91COMPILATIONTAPE	Flat #1	KJ789876 -1300 +01	-1327 +02	27+02	
NAB91COMPILATIONTAPE	Flat #1	KH123456 -5083 +00	-5128 +01	45+02	
NAB91COMPILATIONTAPE	Flat #1	KJ789876 -1393 +08	-1441 +14	48+07	
NAB91COMPILATIONTAPE	Flat #1	KH123456 -5128 +02	-5172 +05	44+04	

Avid Technology, Inc.

Pull list for edl file picture:

Seq	first edge	last edge	roll	Lab Roll	length	scene	take
004	KH123456 -5020 +11	see OPTICAL 2		Flat #1	1+04	1	2
008*	KH123456 -5052 +06	see OPTICAL 3		Flat #1	2+04	2	1
017*	KH123456 -5053 +15	-5057 +11		Flat #1	3+13	2	1
018	KH123456 -5083 +00	-5083 +13		Flat #1	0+14	3	2
013	KH123456 -5085 +05	-5091 +10		Flat #1	6+06	3	2
016	KH123456 -5132 +02	-5142 +04		Flat #1	10+03	3a	1
007	KJ789876 -1236 +02	-1243 +09		Flat #1	7+08	6	1
019	KJ789876 -1244 +09	-1248 +09		Flat #1	4+01	6	1
012	KJ789876 -1305 +03	see OPTICAL 4		Flat #1	1+08	7	1
006	KJ789876 -1332 +01	see OPTICAL 2		Flat #1	7+05	7	2

```
003  KJ789876 -1370 +05          -1372 +05      Flat #1            2+01      9    1

014  KJ789876 -1399 +05          -1409 +08      Flat #1           10+04      9    3

021  KJ789876 -1412 +08      see OPTICAL  5     Flat #1            6+02      9    3

002  KJ789876 -1447 +03      see OPTICAL  1     Flat #1            4+02     10    5

020  KJ789876 -1453 +07          -1464 +11      Flat #1           11+05     10    5

010    LEADER -0000 +00          -0089 +15      35mm LEADER       90+00

015    LEADER -0000 +00          -0003 +14      35mm LEADER        3+15

001    OPTICAL Number 1            FADE IN      EFFECT             1+08

005    OPTICAL Number 2           DISSOLVE      EFFECT             3+00

009*   OPTICAL Number 3           FADE OUT      EFFECT             1+08

011    OPTICAL Number 4            FADE IN      EFFECT             1+08

022    OPTICAL Number 5           FADE OUT      EFFECT             1+08
```

```
Avid Technology, Inc.

Scene Pull List for edl file picture:

Tapename             Lab Roll          first edge            last edge  length  scene
--------             ----------        ----------            ---------  ------  -----
NAB91COMPILATIONTAPE   Flat #1     KH123456 -5019 +11         -5050 +04   30+10

NAB91COMPILATIONTAPE   Flat #1     KH123456 -5050 +05         -5082 +15   32+11

NAB91COMPILATIONTAPE   Flat #1     KH123456 -5083 +00         -5128 +01   45+02

NAB91COMPILATIONTAPE   Flat #1     KH123456 -5128 +02         -5172 +05   44+04

NAB91COMPILATIONTAPE   Flat #1     KJ789876 -1234 +00         -1300 +00   66+01

NAB91COMPILATIONTAPE   Flat #1     KJ789876 -1300 +01         -1327 +02   27+02

NAB91COMPILATIONTAPE   Flat #1     KJ789876 -1327 +03         -1368 +12   41+10

NAB91COMPILATIONTAPE   Flat #1     KJ789876 -1368 +13         -1393 +07   24+11

NAB91COMPILATIONTAPE   Flat #1     KJ789876 -1393 +08         -1441 +14   48+07

NAB91COMPILATIONTAPE   Flat #1     KJ789876 -1441 +15         -1575 +03  133+05
```

```
Avid Technology, Inc.

Negative Dupe list for edl file picture:

               first edge  dupe negative start   scene
Seq             last edge  dupe negative end      take     roll
---            ----------  -------------------   -----     ----
008  KH123456 -5052 +06     KH123456 -5052 +06       2     Flat #1
               -5054 +09    KH123456 -5057 +11       1
017  KH123456 -5053 +15                              2     Flat #1
               -5057 +11                             1
009    OPTICAL Number 3    KH123456 -05054 +10       2     Flat #1
                           KH123456 -05056 +07       1


Avid Technology, Inc.

Optical effects list for EDL file picture:      (5 effects)

Num: 001            Type: Fade-in          Length:   1+08   (24 frames)
Cut: 001
Edl: 001            OUT:                   IN:
                    ---                    --
                    Roll:  BLACK           Roll:  Flat #1
                    Scene:                 Scene: 10
                    Take:                  Take:  5

Scene start:        BLACK
FADE start:         BLACK                  KJ789876 -01445 +11
FADE center:        BLACK                  KJ789876 -01446 +06
FADE end:           BLACK                  KJ789876 -01447 +02
Scene end:                                 KJ789876 -01451 +05

----------------------------------------------------------------

Num: 002            Type: Dissolve         Length:   3+00   (48 frames)
Cut: 005
Edl: 004            OUT:                   IN:
                    ---                    --
                    Roll:  Flat #1         Roll:  Flat #1
                    Scene: 1               Scene: 7
                    Take:  2               Take:  2

Scene start:        KH123456 -05020 +11
DSLV start:         KH123456 -05021 +15    KJ789876 -01329 +01
DSLV center:        KH123456 -05023 +06    KJ789876 -01330 +08
DSLV end:           KH123456 -05024 +14    KJ789876 -01332 +00
Scene end:                                 KJ789876 -01339 +05

----------------------------------------------------------------
```

```
Num: 003            Type: Fade-out          Length:   1+14  (30 frames)
Cut: 009
Edl: 007            OUT:                    IN:
                    ---                     --
                    Roll:  Flat #1          Roll:  BLACK
                    Scene: 2                Scene:
                    Take:  1                Take:

Scene start:        KH123456 -05052 +06
FADE start:         KH123456 -05054 +10     BLACK
FADE center:        KH123456 -05055 +08     BLACK
FADE end:           KH123456 -05056 +07     BLACK
Scene end:                                  BLACK


Num: 004            Type: Fade-in           Length:   1+08  (24 frames)
Cut: 011
Edl: 008            OUT:                    IN:
                    ---                     --
                    Roll:  BLACK            Roll:  Flat #1
                    Scene:                  Scene: 7
                    Take:                   Take:  1

Scene start:        BLACK
FADE start:         BLACK                   KJ789876 -01303 +11
FADE center:        BLACK                   KJ789876 -01304 +06
FADE end:           BLACK                   KJ789876 -01305 +02
Scene end:                                  KJ789876 -01306 +10

------------------------------------------------------------

Num: 005            Type: Fade-out          Length:   1+14  (30 frames)
Cut: 022
Edl: 017            OUT:                    IN:
                    ---                     --
                    Roll:  Flat #1          Roll:  BLACK
                    Scene: 9                Scene:
                    Take:  3                Take:

Scene start:        KJ789876 -01412 +08
FADE start:         KJ789676 -01413 +10     BLACK
FADE center:        KJ789876 -01419 +08     BLACK
FADE end:           KJ789876 -01420 +07     BLACK
Scene end:                                  BLACK

------------------------------------------------------------
```